CLEAN COPY OF THE CLAIMS PROVIDED UNDER 37 C.F.R. 1.121

1. A computerized wagering game apparatus, comprising:

a computerized game controller having a processor, memory, and nonvolatile storage, the computerized game controller being operable to control a computerized wagering game;

an operating system comprising: a system handler application operable to dynamically link with at least one gaming program object;

an Application Program Interface callable from the gaming program object and

an operating system kernel that executes the system handler application.

4. The computerized wagering game apparatus of claim 1, wherein data variables modified by the gaming program objects are stored in nonvolatile storage, and functions verify that the operating system or code for a shared object has not changed.

6. The computerized wagering game apparatus of claim 4, wherein changing a data variable in nonvolatile storage causes execution of a corresponding callback function in the system handler application gaming program object.

13. A method of managing data in a computerized wagering game apparatus via a system handler application, comprising:

loading a first program object and providing an Application Program Interface,

executing the first program object,

storing data variables in nonvolatile storage, such that a second program object later loaded can access the data variables in nonvolatile storage,

D4

unloading the first program object, and

loading a second program object.

---

19. A machine-readable medium with instructions thereon, the medium being within a wagering apparatus, the instructions when executed operable to cause a computer to:

cause a system handler application to load and execute gaming program objects;

cause a loaded gaming program object to call up a library of functions provided by the system handler application;

D5

load a first program object from the library,

execute the first program object,

store data variables in nonvolatile storage, such that a second program object in the library later loaded can access the data variables in nonvolatile storage,

unload the first program object, and

load the second program object.

---

22. A machine-readable medium with instructions thereon, the instructions when executed operable to cause a computer to manage at least one gaming program object via a system handler application, the gaming program object calling up an Application Program Interface such that a single gaming program object is loaded and executed at any one time but gaming program objects are operable to share data via the program variables in a nonvolatile storage.

D6

---

47. A method of modifying an operating system kernel, comprising at least one modification to obtain functionality selected from the group consisting of:

1) accessing user level code from ROM;

2) executing user level code from ROM;

D7

3) zeroing out unused RAM;

4) testing and/or hashing the kernel to verify that the operating system kernel or a code for a shared object has not changed;

D7

5) disabling selected device handlers.

52. The computerized wagering system of claim 1 wherein an Application Program Interface is also dynamically linked from the systems handler.

D8

53. The machine readable medium of claim 19 wherein the instructions when executed are operable to dynamically link an Application Program Interface to a gaming program object.

54. The machine readable medium of claim 22 wherein the instructions when executed are operable to dynamically link an Application Program Interface to a gaming program object.

55. The computerized wagering game apparatus of claim 1 wherein the operating system operates a function to verify that the operating system kernel or a code for a shared object has not changed.

D9

56. The method of managing data in a computerized wagering game apparatus via a system handler application according to claim 1 wherein a function is performed to verify that an operating system kernel or a code for a shared object has not changed.

57. The machine-readable medium with instructions thereon wherein a function is performed to verify that an operating system kernel or a code for a shared object has not changed